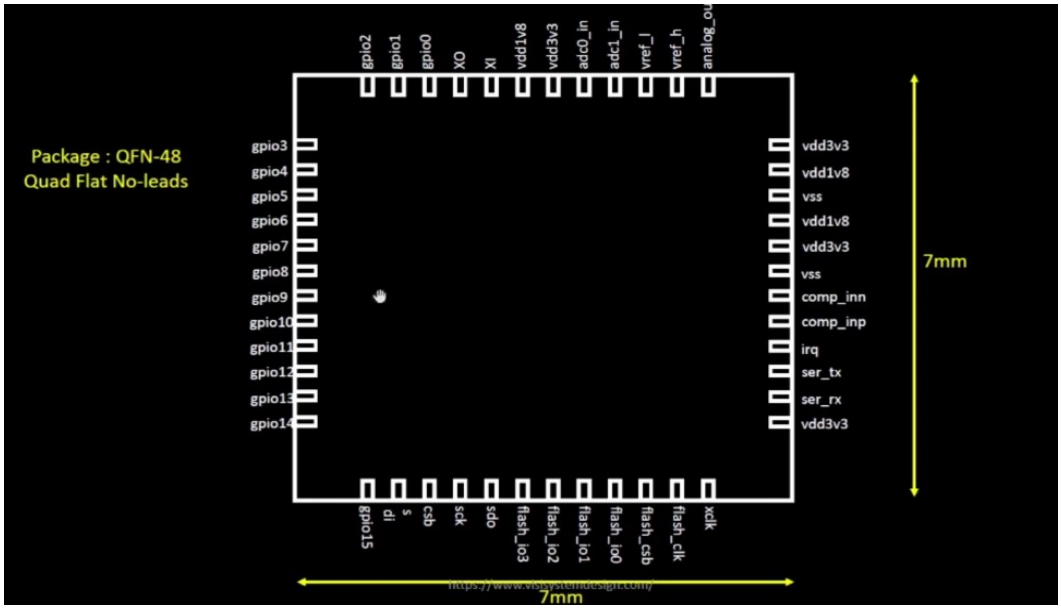
DAY-1:
SKY130_D1_SK1:HOW TO TALK TO COMPUTERS:

SKY_L1:Introduction to QFN-48 Package,chip,pads,core,die and IP

let us know and understand about chip inside a microcontroller.

PACKAGE:it is the protective enclosure and interface for an ic providing electrical connections,thermal management,and mechanical support. Below is QFN-48(quad flat no leads).



Chip is located inside a package.chip is connected to different pins of package. In package,we have

GPIO PINS:

General purpose input output pins inthe package of a chip are versatile digital pins that can be configured by the user to act as either inputs or outputs.

FLASH PINS:

Flash I/O pins refer to the pins on a microcontroller or microprocessor that are specifically designated for interfacing with external flash memory. These pins are used to transfer data between the chip and the flash memory

Pin Functions:

Clock (CLK):

Synchronizes data transfer between the microcontroller and flash memory.

Data In/Out (MOSI/MISO):

Used for sending data to and from the flash memory.

Chip Select (CS):

Activates the flash memory chip that the microcontroller wants to communicate with .Additional Data Lines:

In protocols like QSPI, additional data lines (e.g., IO0, IO1, IO2, IO3) are used to facilitate faster data transfer

Flash csb:

The term "Flash_CSB" refers to the Chip Select Bar (CSB) pin used in flash memory interfaces, particularly in serial communication protocols like SPI (Serial Peripheral Interface) and QSPI (Quad SPI). Here are the details about the Flash_CSB pin:

Function: The CSB (Chip Select Bar) pin is an active-low signal used to select the flash memory chip that the microcontroller or processor wants to communicate with. When the CSB pin is driven low (0V), the corresponding flash memory chip is enabled and can participate in data transfer. When the CSB pin is high (inactive), the flash memory chip is disabled and ignores communication signals on the other lines

XclkFunction:

XCLK serves as an external timing source for the device. It provides a stable clock signal that the microcontroller or DSP can use for its internal operations, such as executing instructions, timing peripheral operations, and driving other clock-dependent processes.

VREF_L (Voltage Reference Low)Function:

The VREF_L pin provides the low reference voltage for the ADC or DAC. This voltage defines the lower limit of the input voltage range that the ADC can convert or the output voltage range that the DAC can produce.

VREF_H (Voltage Reference High)Function:

The VREF_H pin provides the high reference voltage for the ADC or DAC. This voltage defines the upper limit of the input voltage range that the ADC can convert or the output voltage range that the DAC can produce

ADC_0_IN:

The 0th (first) analog input channel of an ADC, used to sample and convert an analog signal to a digital value.

ADC1_IN:

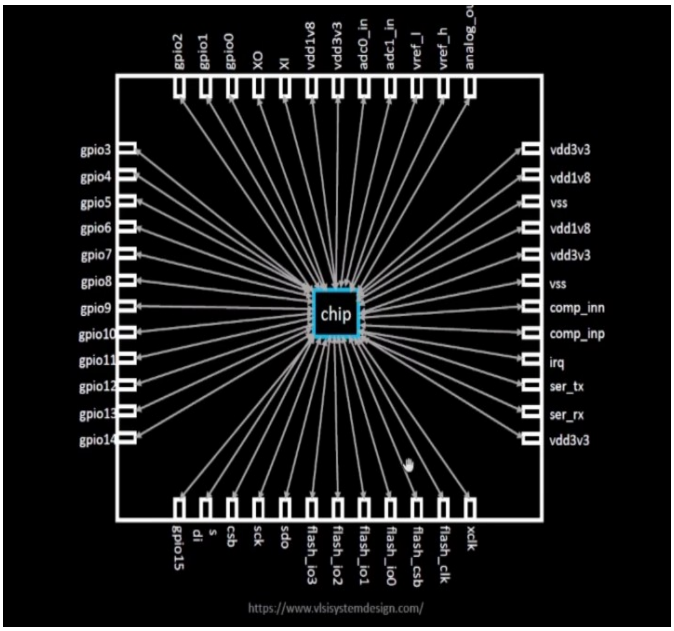
The 1st (second) analog input channel of an ADC, used similarly for a different analog signal.These channels allow the ADC to measure multiple analog signals by converting their voltage levels to digital values that the microcontroller can process.

X0 (XTAL_OUT/OSC_OUT)

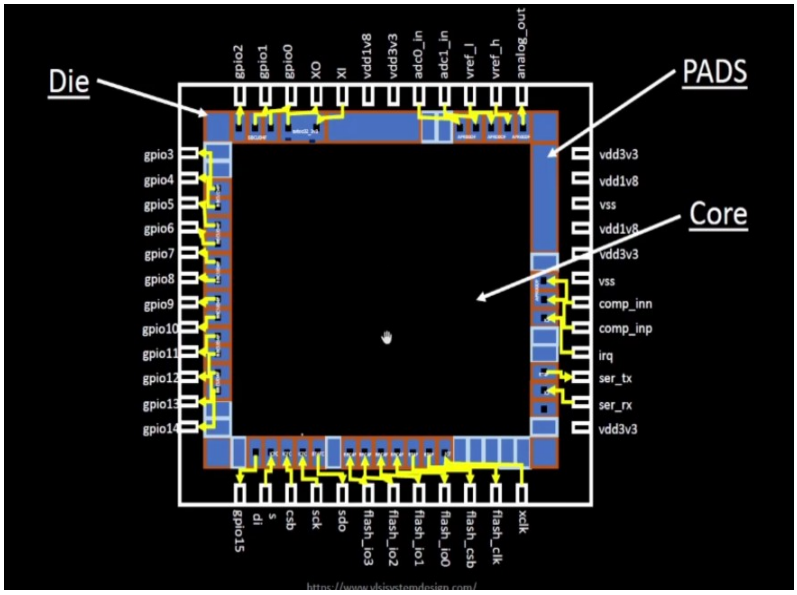
Output pin for the clock signal from the external crystal oscillator.

XI (XTAL_IN/OSC_IN):

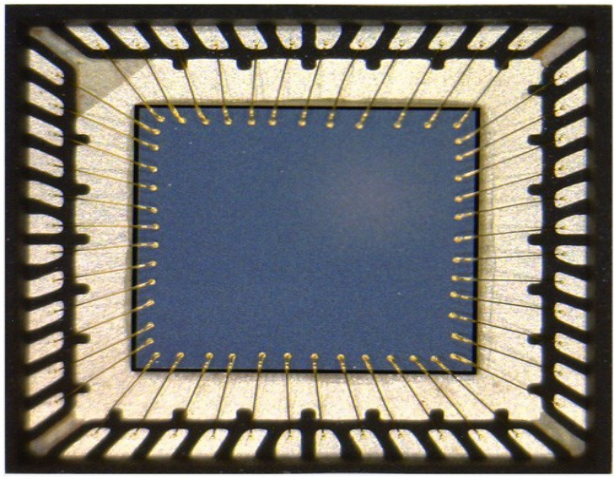
Input pin for the clock signal to the microcontroller from the external crystal oscillator.



There are also some other components which we need to know:



Wire bonds:
Wire bonds are tiny wires used to connect integrated circuit (IC) chips or semiconductor devices to the external leads of the package or to other components on a printed circuit board (PCB).
Wire BondsFunction:
Wire bonds serve as electrical connections between the active elements (such as transistors, diodes, or resistors) within the IC chip and the external leads or terminals of the package.They provide a means for transmitting electrical signals, power, and ground connections between the IC and the rest of the electronic circuit.



Foundry IPs
Foundry IPs are pre-designed and pre-verified functional blocks or modules that are licensed by semiconductor companies for use in their custom IC designs.These IPs are developed by semiconductor foundries, specialized IP vendors, or in-house design teams and are made available for integration into custom SoCs or ASICs.
Types of Foundry IPs
Standard Cells:
These are basic building blocks of digital ICs, consisting of logic gates (AND, OR, etc.), flip-flops, and other fundamental digital circuit elements.
Memory IPs:
These include memory arrays such as SRAM (Static Random Access Memory) and ROM (Read-Only Memory), as well as specialized memory controllers
Interface IPs:
These provide standardized interfaces for connecting different components or subsystems within an SoC, such as USB (Universal Serial Bus), PCIe (Peripheral Component Interconnect Express), HDMI (High-Definition Multimedia Interface), and Ethernet.
Analog and Mixed-Signal IPs:

These include analog-to-digital converters (ADCs), digital-to-analog converters (DACs), analog filters, PLLs (Phase-Locked Loops), and other analog and mixed-signal circuitry.

Process-Specific IPs:

These IPs are tailored to specific semiconductor manufacturing processes and technologies offered by the foundry. They are optimized for performance, power, and area characteristics of a particular process node.

Die:

The "die," also known as the "chip" or "dielectric," is the actual semiconductor material on which the integrated circuit is fabricated.It is a small, rectangular piece of silicon (or other semiconductor material) on which multiple electronic components, such as transistors, resistors, capacitors, and interconnects, are fabricated using semiconductor manufacturing processes.

Function:

The die contains the active electronic components of the integrated circuit, including logic gates, memory cells, analog circuits, and other functional blocks.It is where the primary computational and data processing functions of the IC occur.

Core: The "core" of an integrated circuit refers to a specific functional block or processing unit within the IC.In a multi-core processor, each core is a separate processing unit capable of executing instructions independently and concurrently with other cores.

Function:

Cores execute program instructions and perform computational tasks, such as arithmetic and logic operations, data processing, and control flow operations.In a multi-core processor, multiple cores can work together to execute tasks in parallel, improving overall performance and efficiency.

SKY_L2:-INTRODUCTION TO RISC-V

RISC-V:

RISC-V (pronounced "risk-five") is an open-source Instruction Set Architecture (ISA) based on the principles of Reduced Instruction Set Computing (RISC). It was developed at UC Berkeley and is designed to be simple, modular, and extensible, making it ideal for a wide range of applications, from microcontrollers to supercomputers.

INSTRUCTION SET ARCHITECTURE:

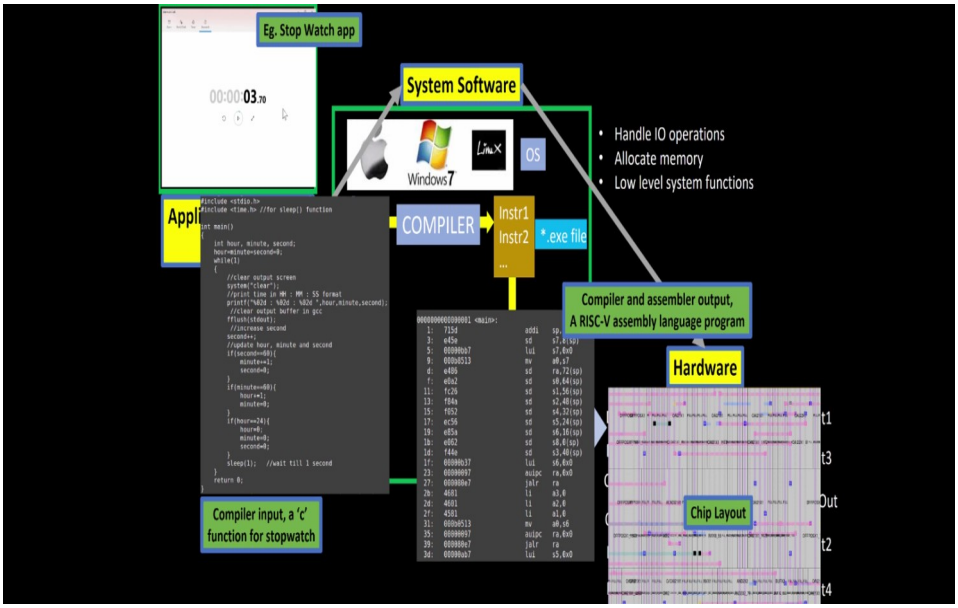
An **Instruction Set Architecture (ISA)** is the interface between hardware and software in a computer system. It defines the set of instructions that a processor can execute and acts as a blueprint for how software communicates with the hardware.

Types of ISA:

1. **RISC (Reduced Instruction Set Computer)**:
    - Simple instructions, fixed-length, optimized for performance (e.g., RISC-V, ARM).
2. **CISC (Complex Instruction Set Computer)**:
    - Complex instructions, variable length, hardware-intensive (e.g., x86).
3. **VLIW (Very Long Instruction Word)**:
    - Executes multiple instructions simultaneously (e.g., Itanium).

INTERACTION BETWEEN HARDWARE AND SOFTWARE:

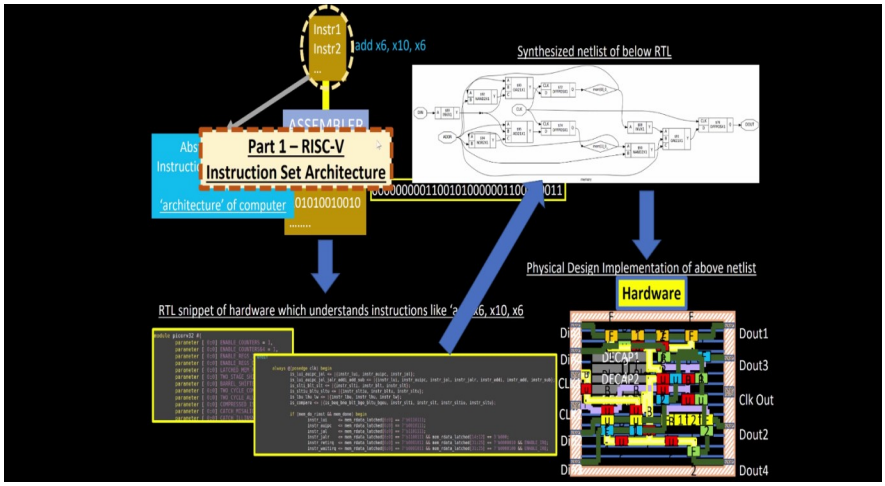
We see that apps run on our laptops,mobile phones etc. They are all hardware. How does this happen?
So,there is an interaction between apps and hardware, i.e system software.
SYSTEM SOFTWARE: System software converts high level programming language like c,c++,java or python to binary level language which is understand by the hardware.



The major components in system software are
1.Operating system
2.Compiler
3.Assembler

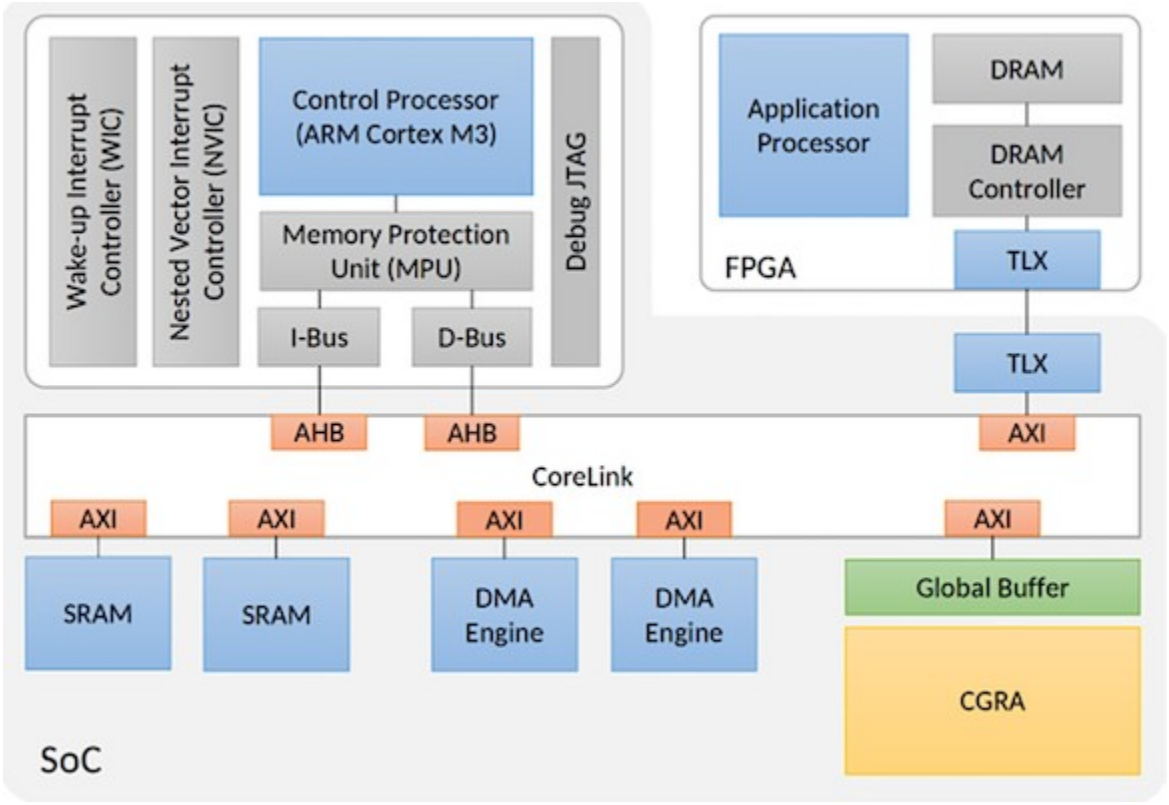
Os generally handles input/output operations,allocates memory, and does low level operations and other part of the operating system converts into assembly language. Apps are written in high level languages like c,c++. they are fed to the compiler. We have a set of instructions. These instructions are dependent on hardware. The hardware may belong to mips,intel etc.
we obtain a *exe files. The obtained *exe files are fed to the assembler.The job of assembler is to convert *exe file into binary language. This is given to hardware, now it generates ouptut.

So, in general, The Instruction set Architecture is fed to the assembler through RISC-V assembly language. We write a RTL(Register Transfer level) snippet which understands instructions. Then we get a netlist for given RTL. It is implemented to hardware.



SOC:
A System on chip(S0C) is an integrated circuit that consolidates multiple components of a complete system into a single chip. It typically includes Processor, memory, input/output interfaces,specialized modules,power management. Soc optmize power,size and performance,making them ideal for compact and energy efficient deisgns.

SOC DESIGN FLOW:
SOC design flow involves several stages to develop a complete system on a chip.Here’s a brief overview:

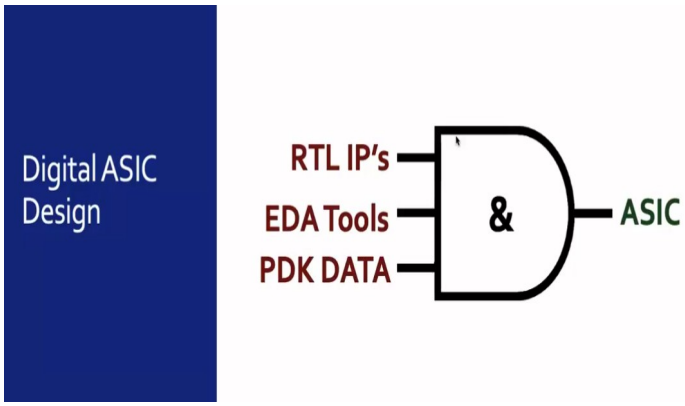


ASIC DESIGN FLOW:

The ASIC (Application-Specific Integrated Circuit) Design Flow is the step-by-step process of designing and fabricating an ASIC .

They require three important key components:

1. Register Transfer level Intellectual properties.
2. Electronic design automation tools.
3. Process design kit Data.



RTL IP'S:

RTL IPs (Register-Transfer Level Intellectual Properties) are reusable pre-designed hardware components described in an HDL (e.g., Verilog, VHDL). They represent functional blocks that can be integrated into larger digital systems during chip design. RTL IPs are a critical part of modern ASIC and SoC (System-on-Chip) design, enabling faster development cycles and reduced effort for implementing complex designs.
They are built once, and can be reused.

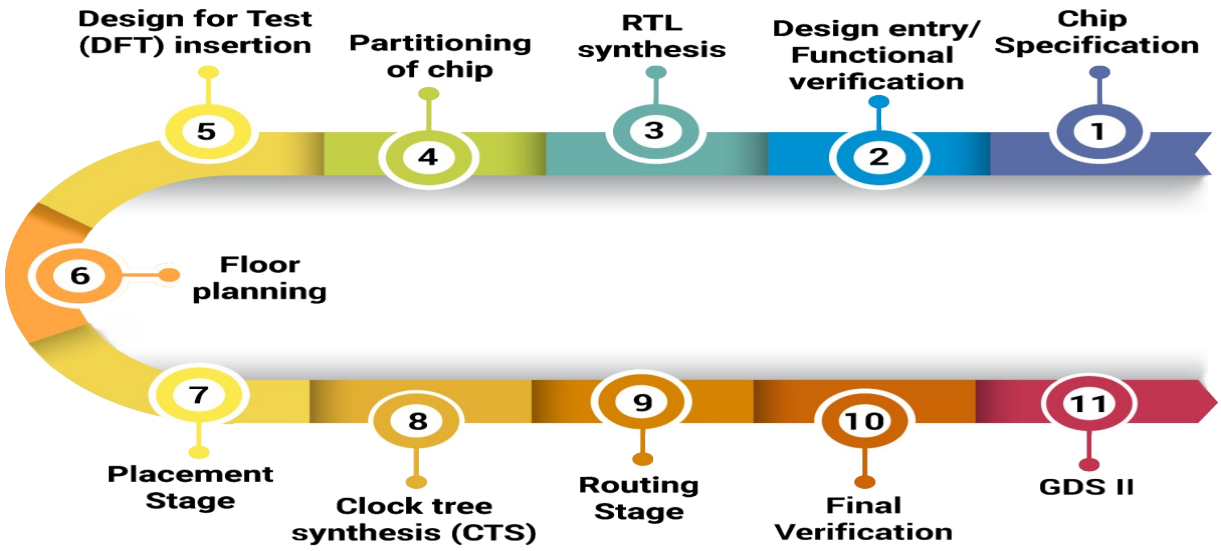
ELECTRONIC DESIGN AUTOMATION TOOLS:

EDA Tools are specialized software platforms used in the design, analysis, verification, and manufacturing of complex integrated circuits (ICs). These tools enable the automation of VLSI design processes, handling the complexity of modern ICs with millions to billions of transistors.
Some EDA tools are Openlane, OpenRoad, Qflow.

PROCESS DESIGN KITS:

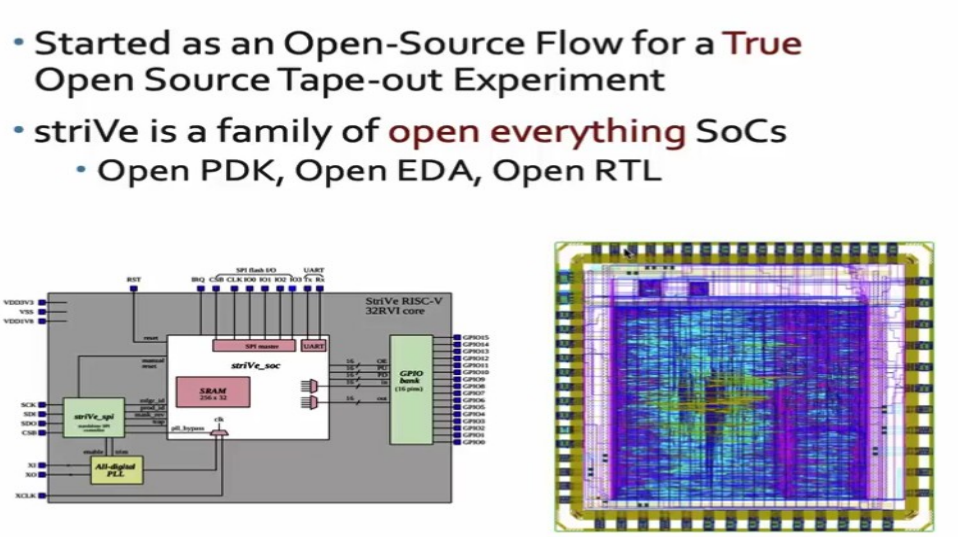
PDKs are collection of files used to model fabrication for the EDA tools to design a IC. It acts as a interface between fabrication units and designers.
The key components of PDK are design rules,technology files,standard cell libraries, SPICE models etc.

DESIGN FLOW:



INTRODUCTION TO OPENLANE AND STRIVE SOC FAMILY

Strive is a family of open everything. For example like open pdk, open eda, open rtl.



OPENLANE:

OpenLane is an open-source, fully automated RTL-to-GDSII (Register Transfer Level to Graphic Database System II) flow for digital ASIC design. It is a part of the OpenROAD project, developed under the DARPA IDEA program, and is actively maintained by the community, including contributions from organizations like Efabless and Google.

OpenLane aims to democratize chip design by providing a platform that integrates open-source tools and methodologies, enabling users to design and tape out custom ASICs with minimal cost..Its main goal is to produce a clean GDS||.. It is tuned for skywater 130nm open pdk.

There are two modes of operation:

1.Autonomous

2. Interactive

AUTONOMOUS MODE OF OPERATION:

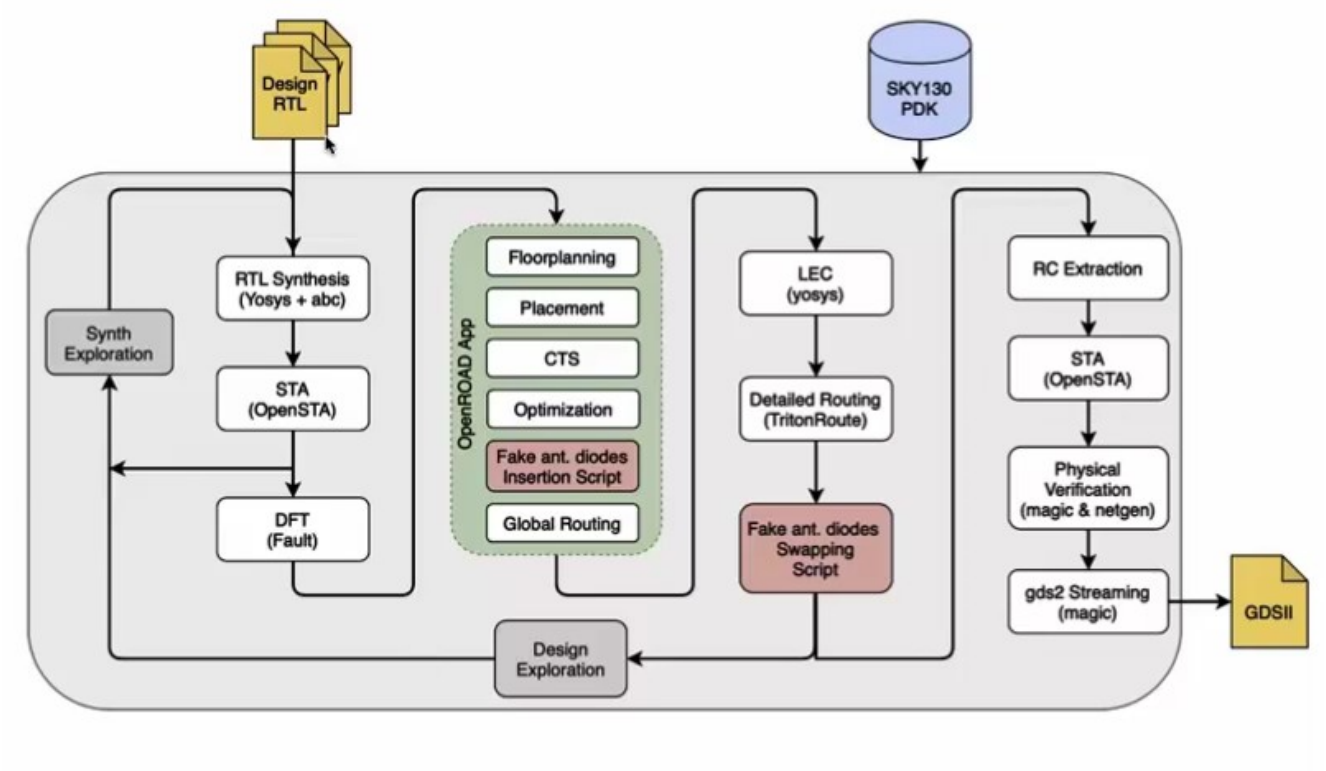
In this mode, OpenLane executes the entire ASIC design flow from start to finish automatically, requiring minimal user intervention.User provide configuration files and input files (e.g., RTL code, constraints) at the beginning, and OpenLane handles all steps, including synthesis, floorplanning, placement, routing, and verification.

INTERACTIVE MODE OF OPERATION:

Provides a step-by-step execution of the design flow, allowing users to control and customize individual steps as needed.In this mode, designers interact with OpenLane via a command-line interface (CLI) to execute specific stages of the flow manually.

Openlane has the best of flow configuration. They have 43 best design configuration

Tools used in openlane flow:



YOSYS: Yosys is an open-source framework for Verilog synthesis. It converts high-level RTL descriptions into gate-level netlists optimized for technology libraries.Supports Verilog input.

- Performs logic synthesis, optimization, and technology mapping.
- Integrates seamlessly with tools like ABC for logic optimization.
- Widely used in open-source ASIC design workflows like OpenLane.

ABC:ABC is a tool for logic optimization and synthesis, specializing in sequential and combinational logic circuits. It operates on gate-level representations to improve circuit performance, area, and power.

OPENSTA: OpenSTA (Open Source Timing Analyzer) is a static timing analysis tool used to verify whether a design meets its timing constraints.

MAGIC: Magic is an open-source layout editor and analysis tool for VLSI design. It is used for creating and verifying IC layouts.

NETGEN: Netgen is an open-source tool used for comparing netlists and performing Layout Versus Schematic (LVS) checks.

KLAYOUT;KLayout is a powerful, open-source IC layout viewer and editor. It supports multiple formats like GDSII and OASIS.

FAULT:: Detects and analyzes faults or defects in the ASIC design layout to ensure reliability and robustness. It performs DFT(Design Foe Test).

FILE FORMATS IN OPENLANE FLOW

1. RTL Files (Verilog/VHDL)(.v) Represents the design in Register Transfer Level (RTL) code.

2.NETLIST FILE(.V): Represents the circuit after synthesis, showing the logical components (gates, flip-flops, etc.) and their connections.

3.CONSTRAINTS FILE:(.tcl): Defines various constraints like timing, placement, and area that must be met during the design flow.

4.DESIGN EXCHANGE FORMAT(.def): Represents the physical design of the circuit, including cell placements, routing, and pin information.

5.STANDARD PARASITIC EXCHANGE FORMAT(.spef): Contains parasitic resistance and capacitance data, which is extracted from the layout to help in accurate timing analysis.

6.Graphic Database System II (.gds): Represents the final layout of the design in a standard format for fabrication.

7.LIBRARY EXCHANGE FORMAT(.lef): Describes the physical attributes of cells in a library, including cell sizes, pin positions, and layer information.

8.LIBERTY FILES(.lib): Contains timing and power models for standard cells (libraries). It is essential for synthesis and timing analysis tools.

9.SYNOPSYS DESIGN CONSTRAINT(.sdc): Specifies constraints for the design, such as clock definitions, input/output delays, and timing exceptions.

10.RC EXTRACTION(.rcs): Represents the extracted resistance and capacitance values for parasitic analysis.

11. REPORT FILES(.rpt OR .log): Contain detailed logs and status reports of each tool executed during the design flow.

12.LAYOUT VS SCHEMATIC(.lvs): Contains results from the Layout Versus Schematic check, ensuring that the layout matches the schematic in terms of connectivity.

13.DESIGN RULE CHECK(.drc): Contains results from the Design Rule Check, which ensures that the layout adheres to the design rules specified by the foundry.

WORKING WITH OPENSOURCE EDA TOOLS:

Openlane consists of various tools.Let’s know about openlane_directory

- Open the Terminal.
- Go to openlane directory using cd Desktop/work/tools/openlane_working_dir

```
vsduser@vsdsquadron:~/Desktop/work/tools$ ls -ltr
total 8
drwxrwxr-x 7 vsduser docker 4096 Jun 28  2021 vsdflow
drwxrwxrwx 5 vsduser docker 4096 Mar 22 15:38 openlane_working_dir
vsduser@vsdsquadron:~/Desktop/work/tools$ ls -ltr
total 8
drwxrwxr-x 7 vsduser docker 4096 Jun 28  2021 vsdflow
drwxrwxrwx 5 vsduser docker 4096 Mar 22 15:38 openlane_working_dir
vsduser@vsdsquadron:~/Desktop/work/tools$ cd openlane_working_dir
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir$ ls -ltr
total 12
drwxr-xr-x  5 vsduser docker 4096 Jun 28  2021 pdks
drwxr-xr-x 10 vsduser docker 4096 Jun 29  2021 openlane_old
drwxr-xr-x 11 vsduser docker 4096 Mar 22 15:38 openlane
```

openlane_working_dir consists of three subdirectories:

1.pdks 2.openlane_old 3.openlane

- Go to **pdks** directory.

Pdks directory consits of all the inofrmation related sky130nm technology. We can see three sub directories: 1) sky130A 2)open_pdks 3)skywater_pdk

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir$ cd pdks
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks$ ls -ltr
total 12
drwxr-xr-x 9 vsduser docker 4096 Jun 28  2021 skywater-pdk
drwxr-xr-x 8 vsduser docker 4096 Jun 28  2021 open_pdks
drwxr-xr-x 5 vsduser docker 4096 Jun 28  2021 sky130A
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks$
```

we work with skywater-pdk. This pdk contains all related lef files, .lib files, timing etc.

Now lets go to the sky130A directory

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks$ cd sky130A
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A$ ls -ltr
total 12
drwxr-xr-x 11 vsduser docker 4096 Jun 28  2021 libs.tech
drwxr-xr-x 14 vsduser docker 4096 Jun 28  2021 libs.ref
-rwxr-xr-x  1 vsduser docker  170 Jun 28  2021 SOURCES
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A$
```

we have 1)libs.tech 2)libs.ref 3) SOURCES

let us see libs.tech

libs.tech: This directory consists of list of tools used in the openlane.

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A$ cd libs.tech
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.tech$ ls -ltr
total 36
drwxr-xr-x  9 vsduser docker 4096 Jun 28  2021 xschem
drwxr-xr-x  2 vsduser docker 4096 Jun 28  2021 xcircuit
drwxr-xr-x  2 vsduser docker 4096 Jun 28  2021 qflow
drwxr-xr-x 10 vsduser docker 4096 Jun 28  2021 openlane
drwxr-xr-x  2 vsduser docker 4096 Jun 28  2021 ngspice
drwxr-xr-x  2 vsduser docker 4096 Jun 28  2021 netgen
drwxr-xr-x  4 vsduser docker 4096 Jun 28  2021 magic
drwxr-xr-x  2 vsduser docker 4096 Jun 28  2021 klayout
drwxr-xr-x  2 vsduser docker 4096 Jun 28  2021 irsim
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.tech$
```

libs.ref:it contains all the process specific files with cell , timing etc.

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A$ cd libs.ref
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref$ ls -ltr
total 48
drwxr-xr-x 10 vsduser docker 4096 Jun 28  2021 sky130_osu_sc_t18
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_ms
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_ls
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_hs
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_hdll
drwxr-xr-x  8 vsduser docker 4096 Jun 28  2021 sky130_fd_pr
drwxr-xr-x  9 vsduser docker 4096 Jun 28  2021 sky130_sram_macros
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_hvl
drwxr-xr-x 11 vsduser docker 4096 Jun 28  2021 sky130_fd_io
drwxr-xr-x  4 vsduser docker 4096 Jun 28  2021 sky130_ml_xx_hd
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_lp
drwxr-xr-x 12 vsduser docker 4096 Jun 28  2021 sky130_fd_sc_hd
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref$
```

now lets go to the **sky_130_fd_sc_hd** directory. (fd-abbrivieated for foundry, sc-standard cell, hd-high density)

- open **sky130_fd_sc_hd** directory and then techlef.

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref$ cd sky130_fd_sc_hd
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd$ ls -ltr
total 88
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 verilog
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 techlef
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 spice
drwxr-xr-x 2 vsduser docker 28672 Jun 28  2021 maglef
drwxr-xr-x 2 vsduser docker 28672 Jun 28  2021 mag
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 lib
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 lef
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 gds
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 doc
drwxr-xr-x 2 vsduser docker  4096 Jun 28  2021 cdl
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd$
```

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd$ cd techlef
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd/techlef$ ls -ltr
total 20
-rwxr-xr-x 1 vsduser docker 18007 Jun 28  2021 sky130_fd_sc_hd.tlef
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd/techlef$ cd sky130_fd_sc_hd.tlef
bash: cd: sky130_fd_sc_hd.tlef: Not a directory
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd/techlef$ vim sky130_fd_sc_hd.tlef

[1]+  Stopped                 vim sky130_fd_sc_hd.tlef
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky130_fd_sc_hd/techlef$
```

techlef files are saved with .tlef extension.

Now go back to the **openlane_working_directory.** And then open **openlane.**

```
(wd now: ~)
vsduser@vsdsquadron:~$ cd Desktop/work/tools/openlane_working_dir/openlane
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$ ls -ltr
total 140
drwxr-xr-x 15 vsduser docker   4096 Jun 29  2021 scripts
-rw-r--r--  1 vsduser docker  20787 Jun 29  2021 run_designs.py
-rw-r--r--  1 vsduser docker   7898 Jun 29  2021 report_generation_wrapper.py
drwxr-xr-x  3 vsduser docker   4096 Jun 29  2021 regression_results
-rw-r--r--  1 vsduser docker  25509 Jun 29  2021 README.md
-rw-r--r--  1 vsduser docker   7273 Jun 29  2021 Makefile
-rw-r--r--  1 vsduser docker  11350 Jun 29  2021 LICENSE
-rwxr-xr-x  1 vsduser docker   6519 Jun 29  2021 flow.tcl
drwxr-xr-x  5 vsduser docker   4096 Jun 29  2021 docs
drwxr-xr-x  5 vsduser docker   4096 Jun 29  2021 docker_build
drwxr-xr-x 44 vsduser docker   4096 Jun 29  2021 designs
-rw-r--r--  1 vsduser docker   1285 Jun 29  2021 CONTRIBUTING.md
-rw-r--r--  1 vsduser docker   5514 Jun 29  2021 conf.py
drwxr-xr-x  2 vsduser docker   4096 Jun 29  2021 configuration
-rwxr-xr-x  1 vsduser docker    966 Jun 29  2021 clean_runs.tcl
-rw-r--r--  1 vsduser docker    709 Jun 29  2021 AUTHORS.md
-rw-r--r--  1 vsduser vsduser   963 May 20  2023 default.cvcrc
drwxrwxr-x  6 vsduser vsduser  4096 Jan 25 07:17 vsdstdcelldesign
```

This directory consists of various files related to the openlane flow like configurations, README.md and design examples and many more.

◆ Now, go to the designs directory.

◆ Here, we will see different design examples, which are already present in openlane and also some new designs are going to be added later.

◆ In these examples, we are going to work with the design picorv32a.

◆ Navigate to the picorv32a directory

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs$ ls -ltr
total 180
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 zipdiv
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 y_huff
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 y_dct
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 xtea
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 wbqspiflash
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 usbf_device
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 usb_cdc_core
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 usb
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 synth_ram
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 sound
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 sha512
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 sha3
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 salsa20
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 s44
-rw-r--r-- 1 vsduser docker 10029 Jun 29  2021 README.md
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 PPU
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 point_scalar_mult
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 point_add
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 ocs_blitter
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 md5
drwxr-xr-x 4 vsduser docker  4096 Jun 29  2021 manual_macro_placement_test
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 ldpcenc
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 ldpc_decoder_802_3an
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 jpeg_encoder
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 inverter
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 genericfir
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 digital_pll_sky130_fd_sc_hd
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 des3
drwxr-xr-x 3 vsduser docker  4096 Jun 29  2021 des
```

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs$ cd picorv32a
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a$ ls -ltr
total 32
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_ms_config.tcl
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_ls_config.tcl
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_hs_config.tcl
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_hdll_config.tcl
-rwxr-xr-x 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_hd_config.tcl
drwxr-xr-x 5 vsduser vsduser 4096 Jan 24 18:15 runs
drwxr-xr-x 2 vsduser docker  4096 Jan 25 16:36 src
-rwxr-xr-x 1 vsduser docker   820 Jan 26 09:49 config.tcl
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a$
```

Here we can see different files inlcuding confid.tcl and src.

◆ config.tcl : Then config.tcl file containsTcl (Tool Command Language) commands that define various settings and parameters crucial for executing the Openlane flow tailored to a specific design project. Within this file, essential design parameters such as the project name, file paths, clock frequency targets, and environmental conditions are typically specified. Additionally, it houses technology-specific details such as the chosen technology node, library paths, and cell specifications necessary for the design process.

◆ Now, go to the src directory as shown below:

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a$ cd src
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/src$ ls -ltr
total 37428
-rw-r--r-- 1 vsduser docker     92423 Jun 29  2021 picorv32a.v
-rw-r--r-- 1 vsduser docker        77 Jun 29  2021 picorv32a.sdc
-rw-rw-r-- 1 vsduser vsduser     1437 Jan 25 07:26 sky130_vsd_inv.lef
-rw-rw-r-- 1 vsduser vsduser 12753932 Jan 25 07:34 sky130_fd_sc_hd__fast.lib
-rw-rw-r-- 1 vsduser vsduser 12732258 Jan 25 07:34 sky130_fd_sc_hd__slow.lib
-rw-rw-r-- 1 vsduser vsduser 12732345 Jan 25 07:34 sky130_fd_sc_hd__typical.lib
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/src$
```

Here, we can see the important files related to the design.

• **picorv32a.v** : It consists the rtl hardware description of the design.

• **picorv32a.sdc**: It contains the variois design constraints like timing constraints, input, output constraints etc.

◆ Now, run the below command, to read the README.md file

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/src$ cd ../../
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs$ cd ../
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$ less README.md

[2]+  Stopped                 less README.md
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$
```

```
 ___  ____  ___  ____  _     ____  ____   ___
/   \ |    \ /  _]|    \ | |    /    ||    \ /  _]
|     ||  o  )  [_ |  _  || |   |  o  ||  _  |  /  [_
|  O  ||   _/    _]|  |  || |___|     ||  |  |  ||    _]
|     ||  | |   [_ |  |  ||     ||  _  ||  |  |  ||   [_
 \___/ |__| |_____||__|__||_____||__|__||__|__|__||_____|

# OpenLANE
[![License](https://img.shields.io/badge/License-Apache%202.0-blue.svg)](https://opensource.org/licenses/Apache-2.0) [![FOSSA Status](https://
pp.fossa.com/api/projects/git%2Bgithub.com%2Fefabless%2Fopenlane.svg?type=shield)](https://app.fossa.com/projects/git%2Bgithub.com%2Fefabless
%2Fopenlane?ref=badge_shield) [![Documentation Status](https://readthedocs.org/projects/openlane/badge/?version=master)](https://openlane.read
thedocs.io/en/master/?badge=master) ![CI](https://github.com/efabless/openlane/workflows/CI/badge.svg?branch=main)

This documentation is also available at ReadTheDocs [here](https://openlane.readthedocs.io/).

## Table of contents
- [Overview](#overview)
- [Prerequisites](#prerequisites)
- [Quick Start](#quick-start)
  - [Installation Notes](#installation-notes)
- [Updating OpenLANE](#updating-openlane)
- [Setting up OpenLANE](#setting-up-openlane)
  - [Pulling the OpenLANE Docker Container](#pulling-the-openlane-docker-container)
  - [Running OpenLANE](#running-openlane)
  - [Command line arguments](#command-line-arguments)
  - [Adding a design](#adding-a-design)
- [OpenLANE Architecture](#openlane-architecture)
  - [OpenLANE Design Stages](#openlane-design-stages)
  - [OpenLANE Output](#openlane-output)
README.md
```

This file contains Various parameters about OPENLANE design flow, openlane directory structure and various commands used in the openlane flow both in Interactive and Autonomous flow.

◆ Now, go back to the openlane directory, now go to the configuration directory.

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$ cd configuration
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/configuration$ ls -ltr
total 64
-rwxr-xr-x 1 vsduser docker  1117 Jun 29  2021 synthesis.tcl
-rwxr-xr-x 1 vsduser docker  1897 Jun 29  2021 routing.tcl
-rw-r--r-- 1 vsduser docker 31784 Jun 29  2021 README.md
-rwxr-xr-x 1 vsduser docker  1288 Jun 29  2021 placement.tcl
-rwxr-xr-x 1 vsduser docker    69 Jun 29  2021 lvs.tcl
-rwxr-xr-x 1 vsduser docker  2358 Jun 29  2021 general.tcl
-rwxr-xr-x 1 vsduser docker  1527 Jun 29  2021 floorplan.tcl
-rwxr-xr-x 1 vsduser docker   808 Jun 29  2021 cts.tcl
-rwxr-xr-x 1 vsduser docker  1113 Jun 29  2021 checkers.tcl
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/configuration$
```

here we have all the tcl files, which are essential in executing openlane flow.

Now lets us know how to open openlane prompt.

- ➔ Open the directory **openlane**
- ➔ type **docker** to launch openlane as below

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir$ cd openlane
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$ docker
bash-4.2$ pwd
/openLANE_flow
bash-4.2$ ls -ltr
total 136
drwxr-xr-x 15 1000  997  4096 Jun 29  2021 scripts
-rw-r--r--  1 1000  997 20787 Jun 29  2021 run_designs.py
-rw-r--r--  1 1000  997  7898 Jun 29  2021 report_generation_wrapper.py
drwxr-xr-x  3 1000  997  4096 Jun 29  2021 regression_results
-rwxr-xr-x  1 1000  997  6519 Jun 29  2021 flow.tcl
drwxr-xr-x  5 1000  997  4096 Jun 29  2021 docs
drwxr-xr-x  5 1000  997  4096 Jun 29  2021 docker_build
drwxr-xr-x 44 1000  997  4096 Jun 29  2021 designs
drwxr-xr-x  2 1000  997  4096 Jun 29  2021 configuration
-rw-r--r--  1 1000  997  5514 Jun 29  2021 conf.py
-rwxr-xr-x  1 1000  997   966 Jun 29  2021 clean_runs.tcl
-rw-r--r--  1 1000  997 25509 Jun 29  2021 README.md
-rw-r--r--  1 1000  997  7273 Jun 29  2021 Makefile
-rw-r--r--  1 1000  997 11350 Jun 29  2021 LICENSE
-rw-r--r--  1 1000  997  1285 Jun 29  2021 CONTRIBUTING.md
-rw-r--r--  1 1000  997   709 Jun 29  2021 AUTHORS.md
-rw-r--r--  1 1000 1000   963 May 19  2023 default.cvcrc
bash-4.2$ ./flow.tcl -interactive
[INFO]:
```

after that, type **./flow.tcl -interactive** so that it starts in interactive mode, we can see the results and reports in each step.

- ➔ We require a package of 0.9
- ➔ for that type **package require openlane 0.9**
- ➔ and then type **prep -design picorv32a**

```
[INFO]: Running interactively
% package require openlane 0.9
0.9
% prep -design picorv32a
[INFO]: Using design configuration at /openLANE_flow/designs/picorv32a/config.tcl
[INFO]: Sourcing Configurations from /openLANE_flow/designs/picorv32a/config.tcl
[INFO]: PDKs root directory: /home/vsduser/Desktop/work/tools/openlane_working_dir/pdks
[INFO]: PDK: sky130A
[INFO]: Setting PDKPATH to /home/vsduser/Desktop/work/tools/openlane_working_dir/pdks/sky130A
[INFO]: Standard Cell Library: sky130_fd_sc_hd
[INFO]: Sourcing Configurations from /openLANE_flow/designs/picorv32a/config.tcl
[INFO]: Current run directory is /openLANE_flow/designs/picorv32a/runs/23-01_13-43
[INFO]: Preparing LEF Files
[INFO]: Extracting the number of available metal layers from /home/vsduser/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.ref/sky13
0_fd_sc_hd/techlef/sky130_fd_sc_hd.tlef
[INFO]: The number of available metal layers is 6
[INFO]: The available metal layers are li1 met1 met2 met3 met4 met5
[INFO]: Merging LEF Files...
ergeLef.py : Merging LEFs
sky130_fd_sc_hd.lef: SITEs matched found: 0
sky130_fd_sc_hd.lef: MACROs matched found: 437
sky130_ef_sc_hd__fill_12.lef: SITEs matched found: 0
sky130_ef_sc_hd__fill_12.lef: MACROs matched found: 1
sky130_ef_sc_hd__decap_12.lef: SITEs matched found: 0
sky130_ef_sc_hd__decap_12.lef: MACROs matched found: 1
sky130_ef_sc_hd__fakediode_2.lef: SITEs matched found: 0
sky130_ef_sc_hd__fakediode_2.lef: MACROs matched found: 1
ergeLef.py : Merging LEFs complete
[INFO]: Trimming Liberty...
[INFO]: Generating Exclude List...
[INFO]: Storing configs into config.tcl ...
```

In this case the **lef** and **tlef** are merged and formed as a single file that is merged .lef

This **merged.lef** file contains the information related to cells and layers.

With this, the design setup is completed.

As, we started the design flow of **picorv32a**, in the **picorv32a** directory, a folder with the name runs is created as shown below indicating the ASIC design flow is started

Now to check, go to **designs** directory and then go to **picorv32a** directory.

In that go **runs** directory. You can see the design you have created.

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane$ cd designs/picorv32a
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a$ ls -ltr
total 32
drwxr-xr-x 2 vsduser docker  4096 Jun 29  2021 src
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_ms_config.tcl
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_ls_config.tcl
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_hs_config.tcl
-rw-r--r-- 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_hdll_config.tcl
-rwxr-xr-x 1 vsduser docker   209 Jun 29  2021 sky130A_sky130_fd_sc_hd_config.tcl
-rwxr-xr-x 1 vsduser docker   444 Jun 29  2021 config.tcl
drwxr-xr-x 3 vsduser vsduser 4096 Jan 23 19:13 runs
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a$ cd runs
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs$ ls -ltr
total 4
drwxr-xr-x 6 vsduser vsduser 4096 Jan 23 19:13 23-01_13-43
```

Now open the directory you see. You can see different directories like tmp, reports and results are created.

In reports directly, we can review reports of every process in openlane flow as shown below.

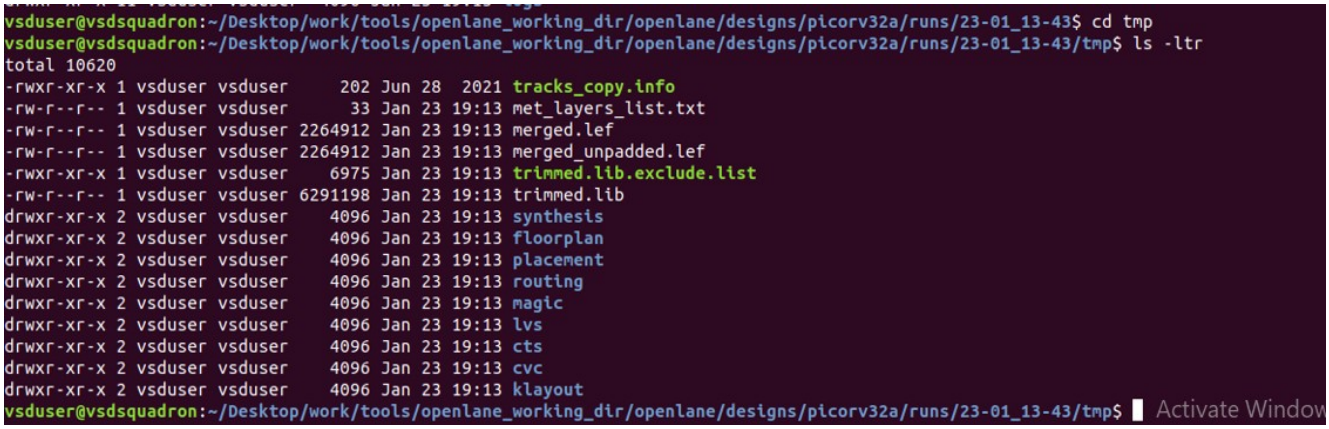
Similarly, in the results directly, we can review the resultant file of every process in the interactive mode of openlane

Here you can see another config.tcl file. This file contains the information about which design parameters are taken. And in this file, we can know, whether the proper execution of every process in Openlane flow, is happenning or not

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs$ cd 23-01_13-43
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43$ ls -ltr
total 52
-rwxr-xr-x  1 vsduser vsduser   170 Jun 28  2021 PDK_SOURCES
drwxr-xr-x 11 vsduser vsduser  4096 Jan 23 19:13 tmp
drwxr-xr-x 11 vsduser vsduser  4096 Jan 23 19:13 results
drwxr-xr-x 11 vsduser vsduser  4096 Jan 23 19:13 reports
-rw-r--r--  1 vsduser vsduser  1585 Jan 23 19:13 cmds.log
-rw-r--r--  1 vsduser vsduser    15 Jan 23 19:13 OPENLANE_VERSION
-rw-r--r--  1 vsduser vsduser 20869 Jan 23 19:13 config.tcl
drwxr-xr-x 11 vsduser vsduser  4096 Jan 23 19:13 logs
```

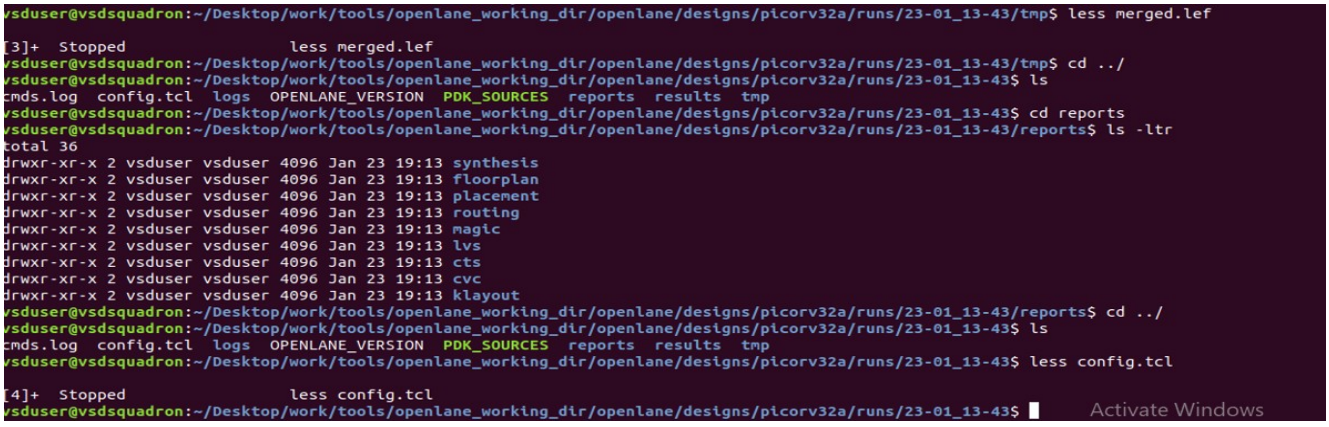
Now go to the **tmp** directory.

Here, we can see the **merged.lef** file, which contains the design parameters like default units for resistance, capacitance etc.



Now you can go to the reports directory. You can see all the reports for synthesis, floorplan,placement etc.

now lets us see **config.tcl** present in the directory that you have created.



```
# Design
set ::env(DESIGN_NAME) "picorv32a"

set ::env(VERILOG_FILES) "./designs/picorv32a/src/picorv32a.v"
set ::env(SDC_FILE) "./designs/picorv32a/src/picorv32a.sdc"

set ::env(CLOCK_PERIOD) "5.000"
set ::env(CLOCK_PORT) "clk"


set ::env(CLOCK_NET) $::env(CLOCK_PORT)




set filename $::env(OPENLANE_ROOT)/designs/$::env(DESIGN_NAME)/$::env(PDK)_$::env(STD_CELL_LIBRARY)_config.tcl
if { [file exists $filename] == 1} {
        source $filename
}
config.tcl (END)
```

also let us see sky130A_sky130_fd_fc_hd_config.tcl

```
# SCL Configs
set ::env(GLB_RT_ADJUSTMENT) 0.1

set ::env(SYNTH_MAX_FANOUT) 6
set ::env(CLOCK_PERIOD) "24.73"
set ::env(FP_CORE_UTIL) 35
set ::env(PL_TARGET_DENSITY) [ expr ($::env(FP_CORE_UTIL)+5) / 100.0 ]

sky130A_sky130_fd_sc_hd_config.tcl (END)
```

There is an order of precendence in openlane to take the values regarding the design requirements.

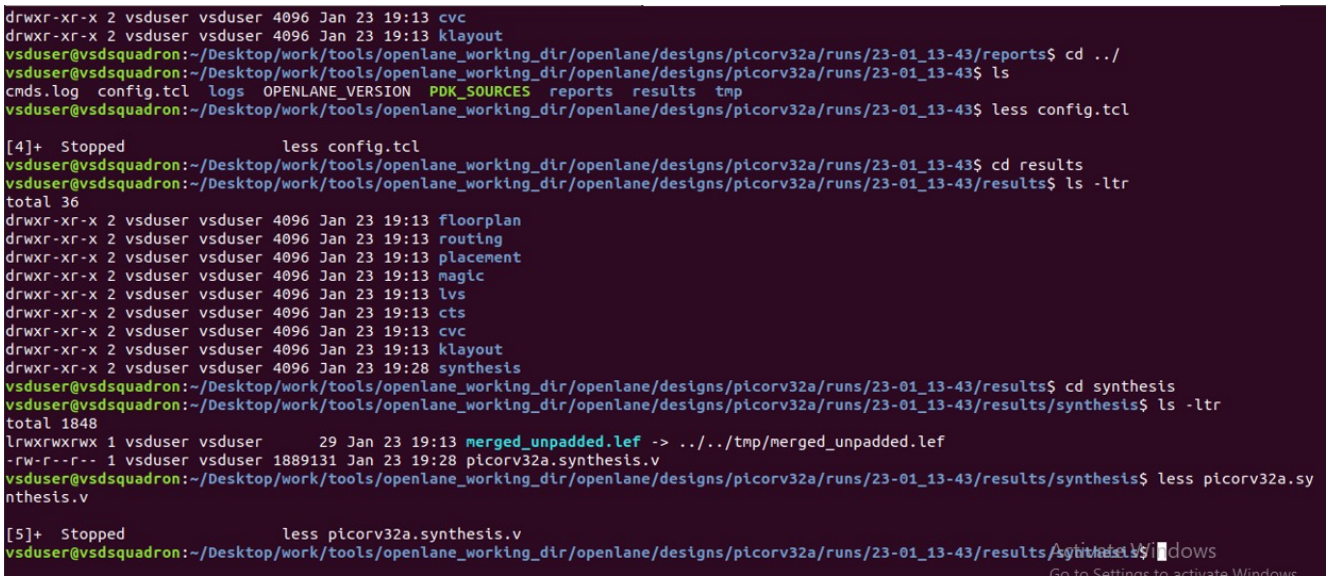
The precedence is as follows: • Default values • Values from **config.tcl** file • Values from **sky130A_sky130_fd_sc_hd_config.tcl** file • **sky130A_sky130_fd_sc_hd_config.tcl** file has the highest precendece and default values are of lowest precendence.

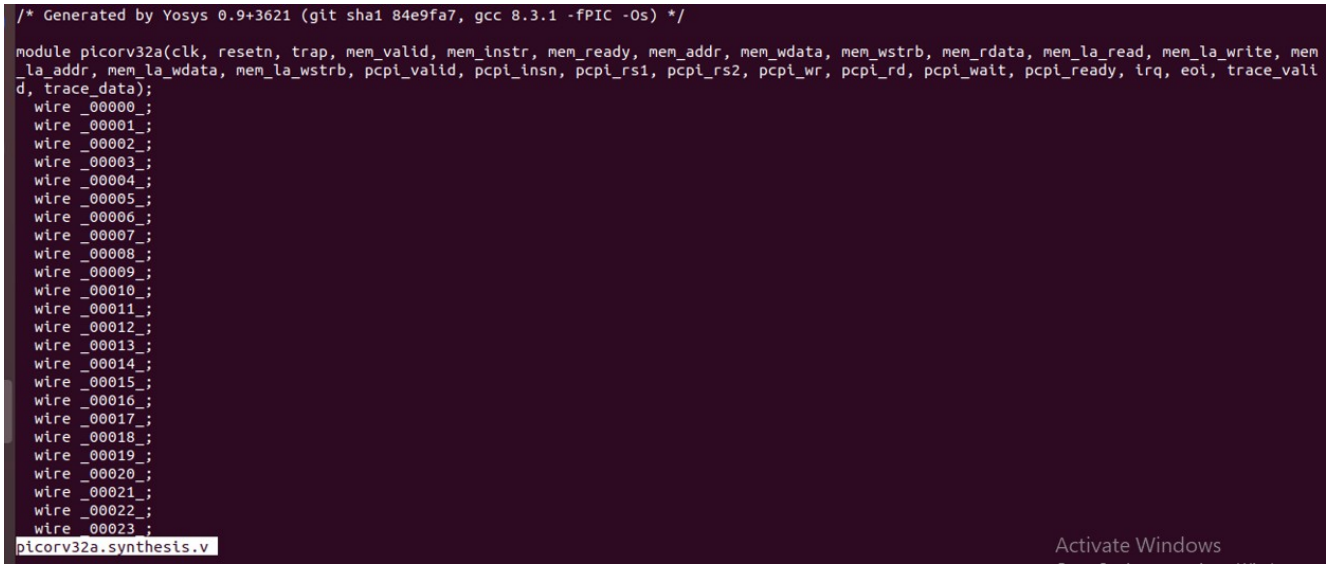
Now, let’s go to the openlane prompt, and run synthesis using **run_synthesis** and we get this

```
set_output_delay $output_delay_value  -clock [get_clocks $::env(CLOCK_PORT)] [all_outputs]
# TODO set this as parameter
set_driving_cell -lib_cell $::env(SYNTH_DRIVING_CELL) -pin $::env(SYNTH_DRIVING_CELL_PIN) [all_inputs]
set cap_load [expr $::env(SYNTH_CAP_LOAD) / 1000.0]
puts "\[INFO\]: Setting load to: $cap_load"
[INFO]: Setting load to: 0.01765
set_load  $cap_load [all_outputs]
tns -759.46
wns -24.89
[INFO]: Synthesis was successful
%
```

Now, lets have the reports and results.

In results, go to the **synthesis** directory.we can see **picorv32a.synthesis.v.**

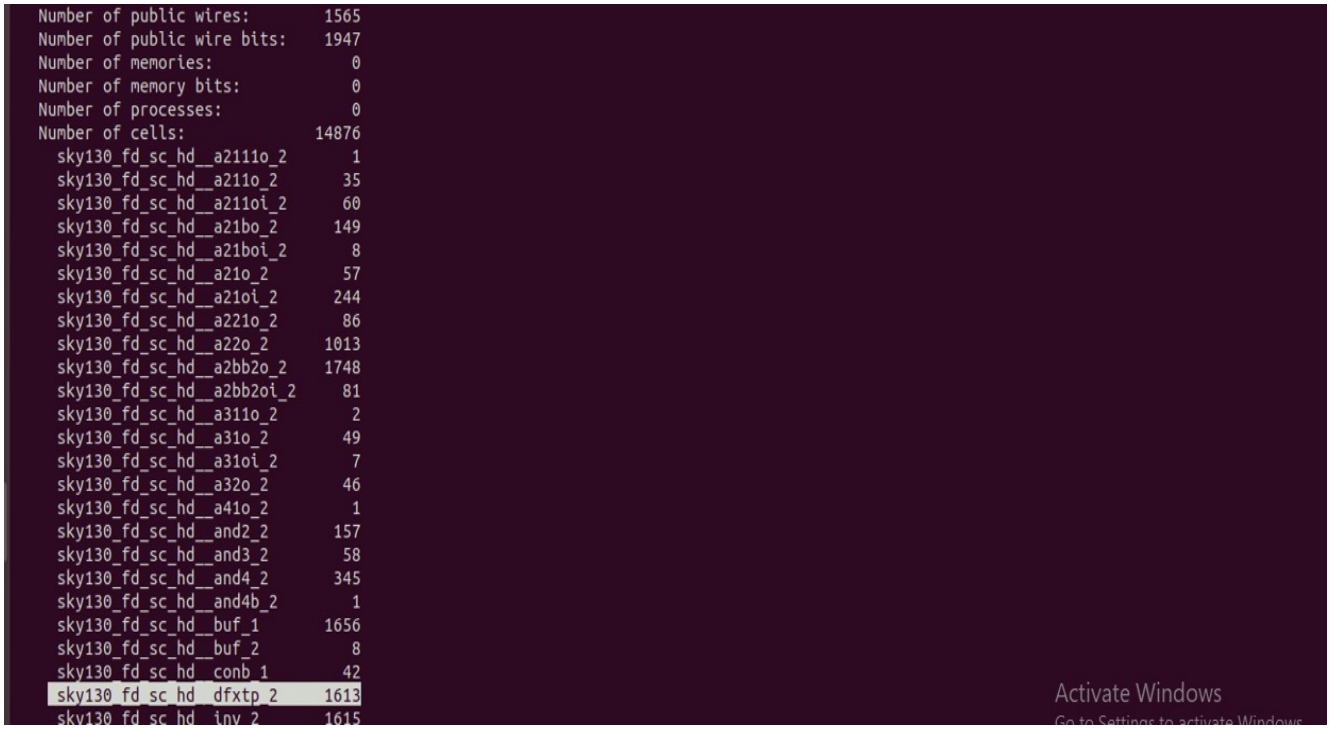




Here, you can see, various reports are generated. Out of these, 1-yosys_4.stat.rpt is the original report of the synthesis process

```
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43$ cd reports
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43/reports$ cd synthesis
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43/reports/synthesis$ ls -ltr
total 1736
-rw-r--r-- 1 vsduser vsduser   1216 Jan 23 19:27 1-yosys_pre.stat
-rw-r--r-- 1 vsduser vsduser    866 Jan 23 19:27 1-yosys_dff.stat
-rw-r--r-- 1 vsduser vsduser  20479 Jan 23 19:28 1-yosys_4.chk.rpt
-rw-r--r-- 1 vsduser vsduser   2674 Jan 23 19:28 1-yosys_4.stat.rpt
-rw-r--r-- 1 vsduser vsduser     12 Jan 23 19:28 2-opensta_tns.rpt
-rw-r--r-- 1 vsduser vsduser     11 Jan 23 19:28 2-opensta_wns.rpt
-rw-r--r-- 1 vsduser vsduser 816771 Jan 23 19:28 2-opensta.timing.rpt
-rw-r--r-- 1 vsduser vsduser  17763 Jan 23 19:28 2-opensta.min_max.rpt
-rw-r--r-- 1 vsduser vsduser 816771 Jan 23 19:28 2-opensta.rpt
-rw-r--r-- 1 vsduser vsduser  74793 Jan 23 19:28 2-opensta.slew.rpt
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43/reports/synthesis$ less yosys_4.stat
.rpt
yosys_4.stat.rpt: No such file or directory
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43/reports/synthesis$ less 2-yosys_4.st
at.rpt
2-yosys_4.stat.rpt: No such file or directory
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43/reports/synthesis$ less 1-yosys_4.st
at.rpt

[6]+  Stopped                 less 1-yosys_4.stat.rpt
vsduser@vsdsquadron:~/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/23-01_13-43/reports/synthesis$
```



Now we can know the flipflop ratio: (number of d flip flopcells)/(number of cells)

flip flop ratio=1613/14876=0.108.